



# CHARTS

Midterm Progress Report

Benjamin Johnson



# Introduction

- Desktop application for creating and sharing musical chord charts / backing tracks
  - *Small music library to handle notes, chords, etc.*
- Cross Platform Support
  - *Linux, Mac, Windows*
  - *Tools:*
    - Qt 6.7 (Qt Widgets UI, QApplication Framework)
    - FluidSynth 2 (MIDI sequencer & playback library)
    - C++17
- Online sharing feature w/ central DB

# Use Case Diagram



The UML class diagram illustrates the architecture of a music application, organized into four main packages: UI (Qt Widgets), Database Interface, Music Lib, and MIDI Handler (FluidSynth Wrapper).

**UI (Qt Widgets) Package:**

- UI StartupDialog:** Contains `newChartSettings : Widgets` and `recentlyOpenedList : Widgets`.
- UI AccountDialog:** Contains `loginForm : Widgets`, `registrationForm : Widgets`, `logoutButton : Widgets`, and `activeUser : User`.
- UI ScorePanel:** Contains `chart : Chart` and `selection : Segment idx`. It has a `selection` association with `Charts Measure` (1 to 1).
- UI MainWindow:** Contains `toolsPalette : Widgets`, `tempChord : WorkingChord`, and `midi : MIDIController`. It has `selection` associations with `Charts Measure` (1 to 1) and `Charts Segment` (1 to \*). It also has a `selection` association with `WorkingChord` (1 to 1).
- UI DB Search Dialog:** Contains `query : SearchQuery` and `searchResultsList : Widgets`. It has a `selection` association with `Charts DB SearchQuery` (1 to 1).

**Database Interface Package:**

- Charts DB User:** Attributes: `userID : uint`, `email : string`, `username : string`, `passwordHash : PassDigest`, `chartEntries : List<ChartEntry>`. Methods: `authenticate()`, `endSession()`, `uploadChart()`, `removeChart()`.
- Charts DB SearchQuery:** Attributes: `name : string`, `authorUserID : uint`, `genre : string`, `resultsList : List<ChartEntry>`. Methods: `authenticate()`, `endSession()`, `uploadChart()`, `removeChart()`.
- Charts DB ChartEntry:** Attributes: `Name : string`, `authorUserID : uint`, `genre : string`, `chartFile : File`, `publicationDate : Date`. Methods: `upload()`, `retrieve()`.

**Music Lib Package:**

- Music TimeSig:** Attributes: `upper : int`, `lower : int`.
- Music KeySig:** Attributes: `tonic : Note`, `scale : enum`, `intervals : bool[intervalCount]`, `sharpKey : bool`.
- Music Note:** Attributes: `value : int`, `flatName : string`, `sharpName : string`. Method: `bool accidental()`.
- Music Chord:** Attributes: `name : str`, `notes : List<PitchedNote>`, `root : Note`, `base : Note`, `quality : enum`, `extension : enum`, `alterations : enum[alterationCount]`, `intervals : bool[intervalCount]`. Methods: `setRoot(Note)`, `setBase(Note)`, `setQuality(quality)`, `setExtension(extension)`, `addAlteration(alt)`, `removeAlteration(alt)`, `bool hasAlteration()`, `bool canAddAlteration()`.
- Music PitchedNote:** Attributes: `value : int`, `octave : int`. Method: `uint : MIDIValue()`.

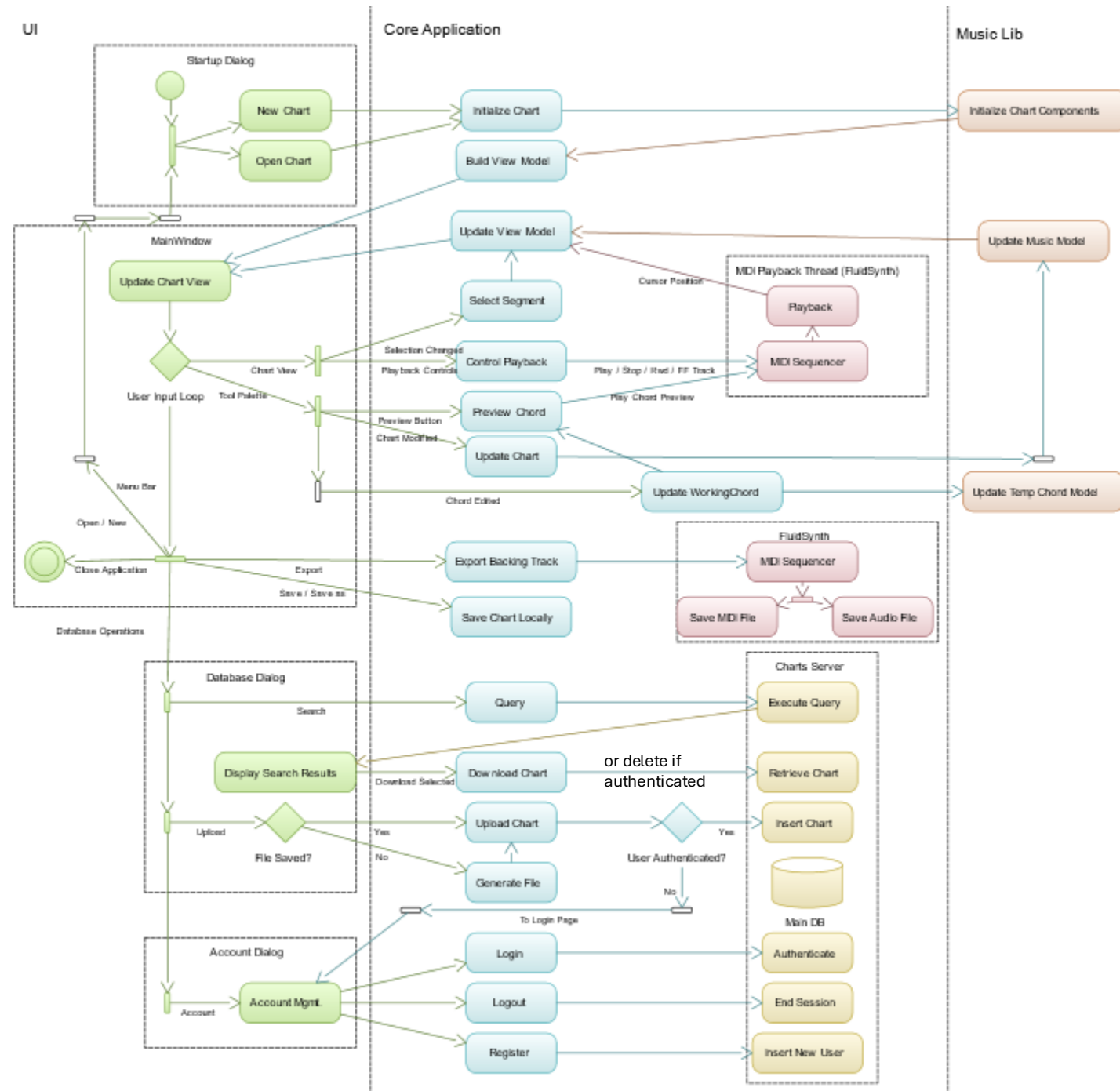
**MIDI Handler (FluidSynth Wrapper) Package:**

- Charts MIDIController:** Attributes: `player : MIDIPlayer`, `playerThread : QThread`, `synth : fsynth`, `settings : fssettings`, `audioDriver : fsaudioDriver`, `sequencer : fssequencer`. Methods: `$ requestPreview(WorkingChord&)`, `$ requestPlayback(Chart&, index)`, `$ requestStopPlayback()`, `$ chordPreviewRequested(Chord&)`, `$ playbackRequested(Chart&, index)`.
- Charts MIDIPlayer:** Attributes: `playbackInterrupted : bool`. Methods: `interruptPlayback()`, `$ previewChord(Chord&)`, `$ playbackChart(Chart&, index)`.

**Associations:**

- UI MainWindow** has a `selection` association with **Charts Measure** (1 to 1).
- UI MainWindow** has a `selection` association with **Charts Segment** (1 to \*).
- UI MainWindow** has a `selection` association with **WorkingChord** (1 to 1).
- UI DB Search Dialog** has a `selection` association with **Charts DB SearchQuery** (1 to 1).
- Charts MIDIController** has a `selection` association with **Charts MIDIPlayer** (1 to 1).
- Music KeySig** has a `selection` association with **Music Chord** (1 to \*).
- Music Note** has a `selection` association with **Music Chord** (1 to \*).
- Music Chord** has a `selection` association with **Music PitchedNote** (1 to \*).

# Activity Diagram



# Completion Progress

- User Interface
  - ~75% - *progress and changes will be made as application code is fleshed out and features can be integrated*
- Music Utils
  - ~99%? - *everything modeled has been implemented; may need to add / change small things*
- Core App
  - ~50% - *UI integration with chord builder & MIDI library complete (it makes sounds!)*
- Server / Database
  - *Planned...*

# Process Model

- Looking more and more like waterfall every day...
- UI & Music Utils are almost done being implemented -> moving to testing stage
- Core Application in implementation stage

# Timeline

## ■ ~~Early October:~~

- ~~—— *Finalize requirements*~~
- ~~—— *UI design & front-end implementation*~~

## ■ ~~Late October:~~

- ~~—— *Front-end refinements*~~
- ~~- *Client-side back-end implementation*~~

## ■ Early November:

- *Finish client-side application*
- *Client-side app testing*
- *Server-side back-end implementation*

## ■ Late November:

- *Finish server-side*
- *Integration testing*



# Challenges

- Cross-Platform Development
  - *Everyone has different problems...*
    - Linking / general inconsistency
    - macOS SDK updates
    - MSVC / “DLL not found”
- Making the “right” choices
  - *Chord representation: interval set vs. pitch set vs. complete enumeration*

# Risk Analysis

- Slow start, but making good progress
  - *Time spent in research phase is paying off*
  - *Should be fully functional by final presentation*
- Focusing on the big picture - Changes from initial plan:
  - *Web app build postponed*
    - Focus on the big 3 desktop platforms
  - *Tutorials may be minimal in the initial release*
- Implementation has been smooth so far